TEC65 USERS MANUAL

INTRODUCTION

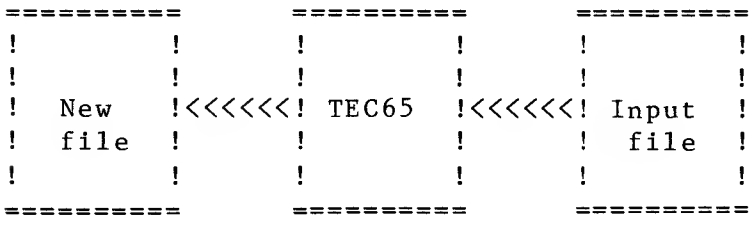
TEC65 is an editing language that is used to create and correct ASCII text files. You can use TEC65 to edit letters, manuscripts, programs and even hexadecimal memory dumps. On the most basic level TEC65 operates as a simple editor. On the most sophisticated level TEC65 is a powerful editing language that allows the users to write programs to control complex editing tasks. Here is a list of some of the more powerful features of TEC65:

String search	Cross buffer search
Search and replace	Wild card searches
Command iteration	Nested iterations
Block move	Ten separate text buffers
Macro commands	Ten command registers
Upper and lower case	Special insert
Device channels	TTY or CRT style console

In TEC65, text is edited using single letter commands. Each command causes the editor to preform a specific editing operation. Several commands can be preformed at once by putting them together into a command string. Single commands and command strings are executed by typing two "ESCAPE" characters in a row. ESCAPE characters (ALTMODE on some keybaords) have special significance to TEC65. Since ALT and ESC are non-printing characters, TEC65 echos a dollar sign "$" for each one. For keyboards that use altmode, TEC65 automatically converts each ALT to an ESCAPE.

GETTING STARTED

In TEC65, a file is edited by passing the text through the editor. The following diagram illustrates the way this happens:



TEC65 reads the old file in and, as changes are made, the edited text is written out to a new file. If the file that you are working on is larger than the computer's memory, TEC65 will work on part of the file at a time. As you finish with one part of the file, TEC65 shuffles the next part of the file into memory and editing proceeds until all of the file is finished. In this way, TEC65 can edit files that are much larger than the computer's memory.

To set up a file to be edited by TEC65, type the following:

```
APX>TEC65 filename.ext
```

This will start TEC65 with the old file for input and the largest empty space on the disk to hold the newly edited file. If you wish to make and edit a brand new file, you should create the new file using the "MAKE" command, then start TEC65:

```
APX>MAKE filename.ext

APX>TEC65 filename.ext
```

Refer to the I-O section of this manual for complete information on handling disk files in TEC65.

BASIC COMMANDS

After you have loaded TEC65, the editor will print a "*" prompting character. This indicates that TEC65 is ready to accept commands. The first commands that we will use are the "INSERT" and "TYPE" commands. These commands are signified by the letters "I" and "T" respectively. The insert command is used to insert text into TEC65'S editing buffer. Here is a sample command string using this command:

```
*IHELLO THERE
$$
```

This string causes TEC65 to insert the text "hello there" into the editing buffer. TEC65 takes each character between the "I" command and the first ESC character and inserts it into the buffer. Since the insert command is terminated only by an ESC character, any number of lines can be inserted into the buffer at once. For example, here is a multi-line insert:

```
*INOW IS THE TIME FOR ALL GOOD MEN
TO COME TO THE AID OF THE PARTY.
$$
```

The "T" command is used to print out all or part of the text in the buffer. Since we need to specify how much of the text is to be printed, the "T" command is generally preceeded by an argument. For example, the letter "H" in front of a "T" command indicates that the whole text should be typed out. Here is an example using our previous inserts:

```
*HT$$
HELLO THERE
NOW IS THE TIME FOR GOOD MEN
TO COME TO THE AID OF THE PARTY.
*
```

The two commands can be combined on the same line like this:

```
*ITHIS IS A TEST
$HT$$
HELLO THERE
NOW IS THE TIME FOR ALL GOOD
MEN TO COME THE AID OF THE PARTY.
```

```
THIS IS A TEST
*
```

Notice that a single ESC character is used to separate the insert characters from the "HT" command, while the two ESCs at the end cause all of the commands to be executed.

If you make a typograpical error while typing commands or text into TEC65, striking the left-arrow key will delete the last character typed. Each additional strike deletes one more character. Each character that is deleted is removed from the screen. Typing Control-R will reprint the corrected line back to the last carriage return. Control-X will delete back to the last carriage return. The entire command string can be deleted by typing Control-Y. Control-G will reprint the entire command string from the beginning.

If it is ever necessary to connect your computer to another terminal, the delete function can be changed to Teletype mode. See the I-O section of this manual.

Since control characters are usually non-printing, TEC65 echos most of them as an uparrow and the character. For example:

```
Control-U  ^U
Control-B  ^B
```

APPLE KEYBOARD

The Apple keyboard does not generate the full ASCII character set. Some of the more important characters omitted are backslash (\) and left bracket ([). The Apple keyboard also doesn't generate lower case characters. To deal with this problem, TEC65 (and APEX) has provisions to generate these characters from the keyboard. Here is a list of characters that can be generated:

KEYSTROKE	PRODUCES
Control-O	\
Control-K	[
Shift-M	]
Control-I	TAB

Lower case is generated through a case lock key. Striking Control-Shift-M places the keyboard in lower case mode. In lower case mode all characters typed are lower case. Lower case characters appear on the screen as inverted characters, black on white. To get out of lower case mode, strike Control-Shift-M a second time.

EXITING

The are several methods of exiting TEC65 back to the operating system. The normal exit is through the "EX" command described in the I-O section of this manual.

Under APEX, most programs abort and exit to the operating system through the Control-C key. In TEC65, Control-C only restarts TEC65. This is useful for aborting long type outs, etc.

Striking Control-P will cause TEC65 to exit to the operating system. This takes you in through the SAVER entry vector so that if the key is accidentally struck there is some chance of recovering any newly generated text.

THE TEXT POINTER

A text pointer is used by the editor to keep track of the users position in the text. For example, after an insert, the text pointer is located after the last character of the insert. Every character inserted into the buffer is inserted at the pointer. Many other commands use this pointer as reference for their operation.

Going back to the "T" command, it uses the text pointer as a reference. For example, the command

```
*T$$
```

causes TEC65 to print one line forward from the editing pointer. If the editing pointer is in the middle of a line, TEC65 prints from the pointer to the end of the line. The command

```
5T$$
```

causes TEC65 to print the next five lines forward from the editing pointer. As you might expect, the command

```
*-5T$$
```

tells TEC65 to print five lines preceeding the editing pointer. To illustrate all of this more clearly, here is what the text buffer might look like if we could look into the computer's memory.

```
============BUFFER============

MARY HAD A LITTLE LAMB
IT'S FLEECE WAS WHITE AS SNOW
AND EVERY WH<p>ERE THAT MARY WENT
THE LAMB WAS SURE TO GO

==============================
```

Here we have four lines of text. The text pointer (represented by <p>) is located in the middle of the word "where". Here are some example commands using this text:

```
*T$$
ERE THAT MARY WENT
*

*-2T$$
MARY HAD A LITTLE LAMB
IT'S FLEECE WAS WHITE AS SNOW
AND EVERY WH*

*-2T2T$$
MARY HAD A LITTLE LAMB
IT'S FLEECE WAS WHITE AS SNOW
AND EVERY WHERE THAT MARY WENT
THE LAMB WAS SURE TO GO
*
```

If you wish to print from the beginning of the line to the text pointer, a zero argument is used:

```
*0T$$
AND EVERY WH*
```

Notice that in TEC65 a zero argument is different from no argument. No argument is the same as an argument of one ("T" is the same as "1T").

```
*0TT$$
AND EVERY WHERE THAT MARY WENT
*
```

CHARACTER ARGUMENTS

Each character in the editing buffer is numbered. If we have 250 characters in our text, then the first character will be 0 and the last will be 249. Also, the text pointer will be located at a numbered location. Several TEC65 commands use these numbered locations as arguments. For example, the "T" command will type out the text between any two numbered positions:

```
*0,5T$$
*MARY *

*5,10T$$
HAD A*
```

Once you have mastered the "T" command, the rest of commands will fall into place since many of them operate in the same manner as "T".

The LINE command is quite similar to the "T" command. It is used to move the text pointer over a designated number of lines. For example,

```
*20L$$
```

moves the text pointer 20 lines forward from it's current position.

```
*-10L$$
```

This command moves the pointer backward 10 lines. Again, TEC65 commands can be combined. For instance,

```
*LT$$
```

moves the pointer one line forward and prints the line.

```
0LT$$
```

This command moves the pointer to the beginning of the current line and prints that line. The commands "-LT, 0LT, LT" are frequently used to step through the text. Since these commands are used so frequently, TEC65 has a special abreviated form of these command strings:

<ESC> or <ALT>	executes ´-LT´
Control-G	executes ´0LT´
line feed (Control-J)	executes ´LT´

These short forms of the commands can only be used if they are the first character typed after the prompt. They cannot be used inside a command string.

The KILL command is virtually identical in format to the "T" command. The "K" command deletes text from the buffer. Where the "T" command prints out certain lines of text, the "K" command deletes the same text given the same arguments. For example

```
*K$$
```

deletes from the text pointer to the end of the line.

```
*5K$$
```

This example deletes five lines forward from the text pointer. If you wish to delete text from the beginning of the line to the pointer use:

```
*0K$$
```

Also, you may use negative arguments to delete backward from the pointer:

```
*-10K$$
```

The "K" command also accepts character number arguments. For example, you can tell TEC65 to delete the 13th through 40th character of a text by typing:

```
*13,40K$$
```

All of the commands that we have discussed so far are line oriented commands. In TEC65, a line is defined as the text between two carriage returns:

```
<CR>MARY HAD A LITTLE LAMB<CR>

    ^--------one line---------^
```

Line feeds are not used as line termination characters inside

the text buffer. This is to avoid the confusion of having two different line termination characters. TEC65 adds a line feed to each carriage return whenever its prints text or sends it to an output device.

CHARACTER ORIENTED COMMANDS

Just as some TEC65 commands deal with lines of text, others deal with characters. The CHARACTER command moves the text pointer a certain number of character positions forward or backward from its current location. For example

```
5C$$
```

moves the text pointer five characters forward, and

```
-12C$$
```

moves the pointer 12 character positions backward.

The DELETE command works identically to the character command. It deletes a certain number of characters forward or backward from the editing pointer. For example

```
*-D$$
```

deletes one character back and

```
*D$$
```

deletes one character forward. Again, all of the commands can be combined into a single command string:

```
*10C-3D0LT$$
```

This string instructs TEC65 to move 10 character positions forward, delete three character backward, then move to the beginning of the line and reprint the line.

SPECIAL ARGUMENTS

TEC65 has several single letter arguments that represent important positions within buffer. For example, the letter "B" represents the beginning of the text buffer. The letter "B" can be used as an argument in character oriented commands. For example

```
*B,26T$$
```

prints from the beginning of the buffer through the 26th character.

```
*B,10K$$
```

This deletes the first ten characters of the text.

A period "." is used to represent the position of the text pointer in the buffer. It may be used as an argument. For example

```
*25,.T$$
```

prints all the text between the 25th character and the current position of the text pointer. The following command deletes everything from the beginning of text up to the text pointer:

```
*B,.K$$
```

The end of text is represented by the letter "Z". It is used in the same manner as other arguments:

```
*.,ZT$$
```

```
*234,ZK$$
```

The argument "B,Z" is frequently used to represent the whole text. In this way the whole text can be deleted:

```
*B,ZK$$
```

Or the whole text printed:

```
*B,ZT$$
```

This construct is used so frequently that TEC65 uses the letter "H" to abbreviate "B,Z". "H" is the whole text argument:

```
*HK$$

*HT$$
```

It is often useful for the user to know the number of characters in the text or the numerical location of the text pointer. The "=" command provides a convienent method of reading these values. This command prints the value of the argument preceeding it. For example

```
*Z=$$
*245
```

prints the number of characters in the text, and:

```
*.=$$
*110
```

prints the position of the text pointer. The example indicates that there are 245 characters in the text and that the pointer is at the 110th character.

The "J" command allows the user to "JUMP" the text pointer to any position within the text:

```
*BJ$$           Jump to the beginning

*ZJ$$           Jump to the end

*26J$$          Jump to the 26th character

*J$$            also Jumps to the beginning
```

SEARCH COMMANDS

TEC65 has the ability to search through the text buffer for strings of characters. Searches are very rapid; TEC65 can find a string of characters at the end of a 16k buffer in less than a second. Searches proceed from the text pointer forward. If TEC65 is able to find the string, the text pointer will be left pointing at the end of the string. If TEC65 fails to find the string it will print an error message and leave the pointer where the search started. The letter "S" commands TEC65 to search for a string of characters. The string begins with the first character after the "S" and ends with the first ESC character. For example

```
*SAUTOMOBILE$$
```

searches for the word automobile. Any character can be included in the search string except ESC, Control-R, Rubout, and Control-X. Generally, you will want TEC65 to print the line when it finds the character string:

```
*SNUMBER 1$OLT$$

*STWELVE O'CLOCK$OTT$$
```

If the first example finds a match, the pointer will be moved to the beginning of the line and the line will be printed. If the second example finds a match, the pointer is left at the end of the string and line is printed.

If a search command is preceeded by a numerical argument, TEC65 will search for the nth occurrence of the string. For example:

```
*5SAPRIL$$
```

will search for the fifth occurrence of the word APRIL.

The character Control-W (^W) is used as a wild card character in a search string. Control-W will match any character. Example:

```
ST^WO$$
```

This search would match "TOO","TWO","TAO" etc.

TEC65 has a special form of search command which searches for one string of characters and replaces it with another string. The letters "FS" call this command:

*FSDOGCAT

This example searches for the first occurrence of the word "DOG" and replaces it with "CAT". The strings do not have to be the same size; in fact, if you don't use the second string, TEC65 will search for the first string and delete it:

*FSBIRDS$$

This example deletes the word "BIRDS" from the text. The "FS" command works in the same manner as the search command. Control-W may be used as a wild card character, and a numerical argument will cause it to search for the nth occurrence of the string.

ITERATIONS

One of the most powerful features of TEC65 is it's ability to execute a command string repeatedly. Part or all of string may be repeated by enclosing the command string in angle brackets, "<>". The number of executions is controlled by the argument preceeding the bracket. For example:

```
*5<LIFROG$0LT>$$
```

The command string inside the brackets moves one line forward, inserts the word "FROG" and reprints the line. Since the string is enclosed in brackets, it is repeated five times. Commands may executed up to 65,000 times. If no argument is used in front of a pair of brackets, the commands are executed infinitly. Most commands will terminate either with an error, or at the end of the buffer. Some commands, like "T", will never terminate, so the editor must be interrupted and the restarted (Control-C). The infinite argument makes global searches and replacements possible. For example, you may print every occurrence of a certain text or replace every occurrence of one string with another string:

```
*J<SHOUSEWIFE$0TT>$$

*J<FSMEN$MEN AND WOMEN$0TT>$$
```

The first command string will print every line that contains the word "HOUSEWIFE". The second command string will replace the the word "MEN" with the words "MEN AND WOMEN" throughout the text. When a search command fails within brackets, TEC65 continues executing the rest of the command string, then exits the loop. At times it is important that no other commands within a loop are executed after the search fails. A semicolon anywhere within the iteration will force an exit at that point after a failure.

If necessary, brackets may be nested:

```
*20<IABC$4<ICDE$>>$$
```

Q-REGISTERS

TEC65 has ten general purpose registers called q-registers. The registers are numbered 0-9 and each register can be used to hold either commands strings or blocks of text.

Text can be scooped into any q-register using the "X" command. The "X" command operates in the same manner as the "T" command. The same lines that would be printed with a "T" command are scooped into the q-register. The number following the "X" indicates the Q-register used. For example:

```
*2X4$$
```

scoops the next two lines of text into q-register number four. The following example scoops the whole text buffer into register nine:

```
*HX9$$
```

Q-register size is dynamically allocated, so that a q-register will expand to take up all of memory if necessary.

Text that has been stored in a q-register is retrieved using the GET command. The "G" command gets the contents of the specified register and inserts it into the text at the current pointer position.

```
*G5$$
```

The act of getting text from a q-register does not destroy the stored text, so that copies of the same text can be inserted at several locations in the text. The act of scooping text into a register always destroys any old content.

The ability to carry text in registers allows TEC65 to rearrange paragraphs, sentences and chapters easily. The registers can also be used to carry frequently used words, phrases or assembly code for insertion when needed.

COMMAND STRINGS

Strings of ordinary TEC65 commands can be loaded into q-registers and executed as needed. Commands may be loaded into the registers in two ways: the first via the "X" command and the second via the "*" command. The "*" command puts the last sucessfully completed command string into a specified register. To use the "*" command, the first character typed on the keyboard after the sucessful command must be a star, otherwise TEC65 throw away the old command string. Here is an example:

```
*SFROG$4D$$
**3$$
```

Here we execute a search and delete command in the usual way. After the command has completed, we type a star followed by the register number. The command string is now stored in register three and may be executed at any time.

The commands in q-registers are executed using the "M" command. For example "M1" executes the commands in q-register number one. The commands are not destroyed when executed and can be executed over and over.

When a command string is inserted directly into the editing buffer and then scooped into a q-register, the special insert must be used to insert the escape characters that separate the commands.

MACROS

One of the uses for the q-registers is executing complex command strings. The strings can be elaborate tasks such as converting one assembler format to another. Strings can be stored on a mass storage device and used when they are needed. You load them into the text area from disk and scoop them into q-registers using the "X" command. Here are two examples of macros:

```
58CS $-CI
$D$-LT$$

FS
$ $0LT$$
```

It frequently happens that inserting characters into a line makes the line too long. The first macro converts a long line to a line about 64 characters long. Executing this macro will make the next line too short. The second macro removes the old carriage return from the line. These two macros can be used in succession to justify a section of text.

DISK I-O

TEC65 has six commands that are used to access and control disk files. The commands control the way in which information is read and written on the disk. In TEC65, a file is edited by passing the text through the editor. TEC65 reads the old file in and, as changes are made, the edited text is written out to a new file. If the file that you are working on is larger than the computer's memory, TEC65 will work on part of the file at a time.

The "EO" opens the input and output devices for interaction with TEC65.

The "EA" command appends text from the input device into the editing buffer. The first append fills the buffer leaving 2k of editing free space. Each subsequent append brings in 256 bytes of text.

The "EW" command writes the current text buffer to the output file. The text buffer is not deleted.

The "EG" command writes the current buffer to the output file, deletes the current buffer and gets the next buffer of text.

The "EC" closes all input and output files.

The "EX" command moves all remaining text from the input file to output, closes files and exits.

Most of the time EO, EG, EX are the only commands used to move through a text file. First, the EO command is used to open and initialize the input and output files. Then, the EG command is used to shuffle the next segment of text into memory. The EG command is executed repeatedly each time you wish to edit the next segment of text. When you reach the end of the file, TEC65 will print an "END OF FILE" message. When you have finished editing this last segment of the file, you can end the editing pass by using the EX command. This will finish moving all of the text into the output file and re-enter the operating system. The EX command can be used at any time to finish editing and re-enter Apex.

CONSOLE I-O

Since TEC65 can be used on both Teletype like devices and CRT like devices, a command has been provided to change the console interface to accommodate either device. When TEC65 is in CRT mode, characters that are deleted disappear from the screen. When it is in TTY mode, the deleted characters are echoed. This is because TTY like devices cannot backspace and erase characters from the paper. In TTY mode, the line delete character also changes from Control-X to Control-U.

EVT$ This command set the console to TTY mode.

EVC$ This command sets the console to CRT mode.

CONSOLE I-O CHANNELS

TEC65 usually communicates through APEX device 1. However it has the ability to switch it's console I-O to other devices. For example, it may be useful to connect an external terminal to the Serial Communications card, so that you can edit on a remote terminal. The standard I-O devices are described in the operating system manual. The following commands to TEC65 are used to change I-O devices:

ED n Change input and output to device n.

EDI n Change input to device n.

EDO n Change output to device n.

SPECIAL INSERTS

Normally, certain characters cannot be inserted into the text. These characters include, Rubout, ESC, ^X, etc. TEC65 has a special form of insert that allows any character, including those not available on the keyboard, to be inserted into the text buffer. If the normal "I" command is preceeded by a numerical argument, the number is converted to ASCII and inserted into the text. For example

27I$$

65I$$

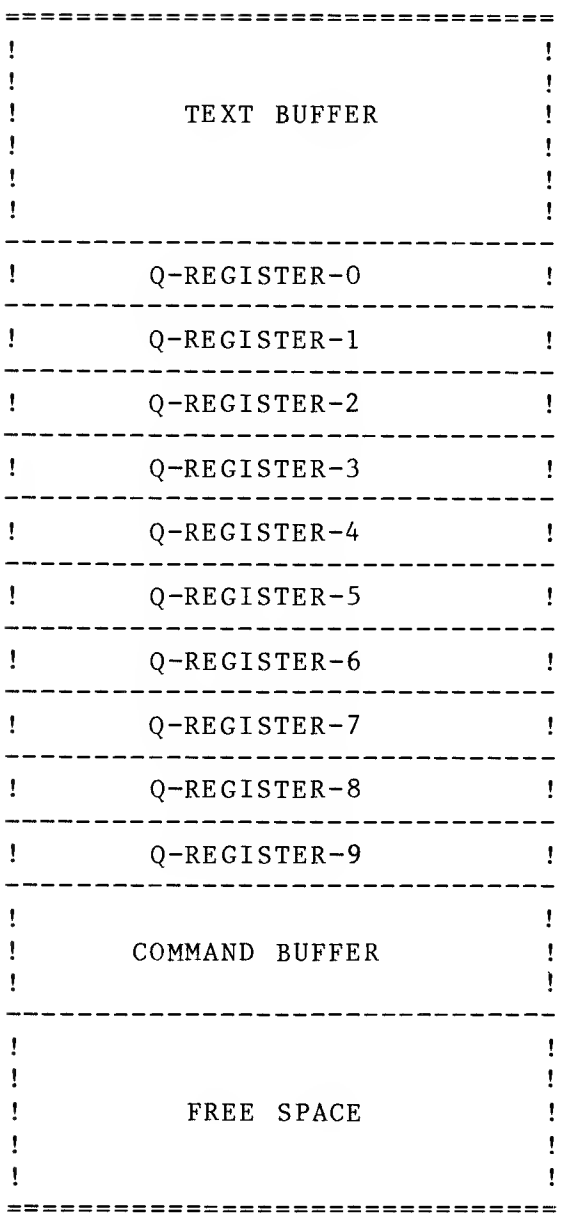
the first example inserts escape and the second example inserts an "A".

ERROR MESSAGES

There are a number of faulty, illegal or dangerous command sequences that can be created in TEC65. The editor checks for most of them and protects the user from their consequences. If an error condition is discovered, the operation is immediately aborted and an error message is printed.

Most of the error messages are self explanitory. Some of the range errors may be confusing when you get them because the buffer is empty. For example, if you execute "HT" when the buffer is empty you will get a message stating that the second argument must be greater than the first. This occurs because when the buffer is empty the argument "H" actually represents "0,0".

MEMORY MAP



All of TEC65's buffers and registers are dynamically allocated. Each area may expand and contract as needed. The only limitation is the amount of memory available in the users system.

SUMMARY OF COMMAND AND SPECIAL CHARACTERS

C	Move a specified number of characters forward or backward.
D	Delete a specified number of characters forward or backward. With a double argument, delete from one character position to another.
FS	Search for a character string and replace it with another.
G	Get the text in a specified q-register and insert it.
I	Insert a text string.
J	Jump to a specified position in the buffer.
K	Kill a specified number of lines forward or backward.
L	Move a specified number of lines forward or backward.
M	Execute the commands in a specified q-register.
T	Type a secified number of lines forward or backward. With a double argument, type the text between one character position and another.
X	Scoop a specified number of lines, forward or backward, into a q-register. With a double argument, scoop the text between one character position and another.
=	Print the value of the preceeding argument.
S	Search for a character string.
<	Start of iteration.
>	End of iteration.
;	Exit from iteration upon search failure.
N	Search to the end of the buffer, write the buffer,

delete it, append and continue search.

* Places the last successful command string into a specified q-register.

I-O COMMANDS

EO Open input and output files.

EA Fill buffer from input file or device.

EC Close input and output files.

EW Write current text buffer to output file or device.

EG Write current buffer, delete current buffer and append.

EX Move the remaining input to the output file and close all files.

EV Set console characteristics, C=CRT, T=TTY.

ED Set devices.

EDI Set input device.

EDO Set output device.

SPECIAL CHARACTERS

. Equivalent of current text pointer position.

H Same as "B,Z". It is an argument indicating the whole text buffer.

B Indicates the beginning of the text buffer.

Z Indicates the end of the text buffer.

, Seperates arguments in a double argument.

<ESC> The escape character has three functions:

1) terminates strings in commands.
2) executes a "-LT" when it is the first character after the prompt character.
3) two escapes execute the current command string.

^G Executes a "0LT" when it is the first character typed.

<LF> Line feed executes a "1LT" when it is the first character typed after the prompt.

^X Deletes current command or text line back to the last carriage return.

^Y Deletes all of the current command or text string.

^W Wild card character in search strings.

^R Reprints the current command line from the last carriage return.

^I Tab character.

^Z Standard end of file character. TEC65 places one at the end of every file it writes and stops reading when it receives one on input.